This document has not yet been approved by the Government for general use or distribution.

Appendix A

Notes

The following material is presented to aid in understanding the object oriented information provided in this presentation. This information is a subset of Appendix N of the ECS System Design Specification. For additional information associated with the object oriented approach, consult the ECS System Design Specification.

N.1.   Object Oriented "Point of View"

The Object Oriented paradigm takes a modeling point of view. The analysis and design phases remain distinct, however a single model is developed (matured) across the phases. The model is constructed by viewing the problem domain as a set of interacting entities and relationships (associations) among the entities. The information developed during the analysis phase(s) becomes an integral part of the design. That is, the analysis model matures into the design model. Each entity (object) is fully described by its data (attributes) and its behavior (operations).

The pieces produced by the Object Oriented paradigm are entity descriptions (classes). The classes relate directly back to the original problem description or represent conceptual entities which can be have distinct meaning in the problem domain.

N.2.   Object Oriented Concepts

Object Oriented Methodology is a development paradigm which organizes a system as a collection of objects, each of which contains both the data and behavior of an entity that has meaning in the context of the development domain.

There are many different methodologies which have been developed to formalize the application of Object Oriented technology. The methodology used on the ECS Program is the Object Modeling Technique (OMT). The methodology provides a formalization of an analysis, design, and development process which is supported by CASE tools, and which supports the entire life-cycle of the program.

N.3 OO Terminology

It is necessary to understand some basic Object Oriented concepts in order to review this presentation. This section defines the high-level concepts.

N.3.1   Object

A concept, abstraction or thing with crisp boundaries and meaning for the problem domain at hand. An object is characterized by a unique name, distinct properties, and well defined behavior.

For example, a gyroscope is an object. It can be uniquely identified by its name, it has attributes and specific behavior that make it the type of instrument that it is.

N.3.2 Class

A group of objects with similar properties (attributes), common behavior (operations), common relationships with other objects, and common meaning.

While a gyroscope is an object, it belongs to a class of instruments which exhibit common attributes (e.g. voltage, cal. factor, dynamic range) and operations (e.g. turn on, turn off, scan). Each of the specific objects which comprise the class is termed an instance of the class, i.e an object is always an instance of a class.

N.3.3 Attributes

A named property of a class describing data values held by each object of the class. Classes describe the data property. Each object holds a value for each attribute defined for the class of which it is a member.

For example, the class instruments might have attributes voltage, calibration factor, and dynamic range. The gyroscope is an object of the class instruments and might have values of one volt, one-degree of deflection per telemetry count, and -180-to+180 degrees of deflection, respectively. Every object belonging to the class would have their own unique values for the same attributes.

N.3.4 Operations

A function or transformation performed on or by the class. The sum of the operations define the common behaviors of all members of the class. All instantiations (objects) of the class share the same operations. In fact, when objects are represented in the model notation the operations are not even included with the object symbology; only class name and the attribute values are provided for objects.

N.3.5 A Class Diagram

Figure N.3.5-1. shows a class diagram with an instantiated object.

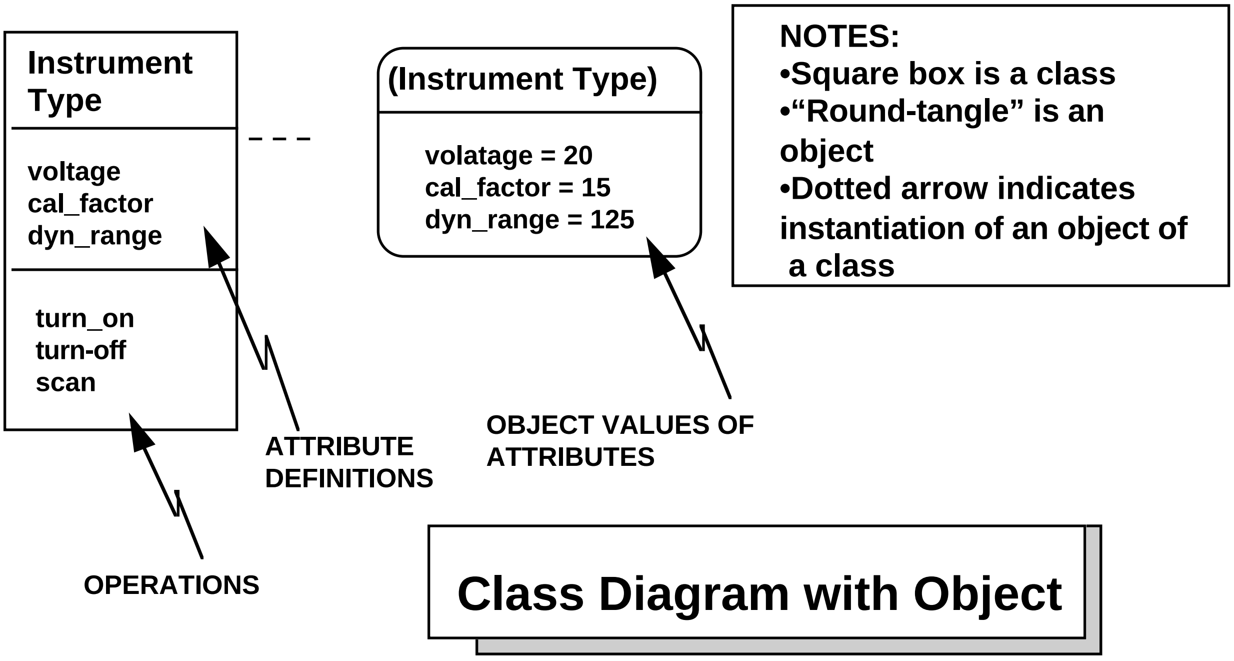


Figure N.3.5-1. Class Diagram with Object

The class is represented by the square box. The class is partitioned into three areas:

- The top area contains the name of the class.
- The middle area lists the attributes of the class. Note that there are no values allocated to the attributes. This is because the attributes in the class are the definition of the attributes, not the actual values. The values are what distinguish one object/instantiation of the class from another.
- The bottom area lists the operations that define the behavior of this class.

The object is represented by the "round-tangle". The object is partitioned into only two areas:

- The top area is, once again, the name of the object is instantiated which has been instantiated from the class. By convention the name is in parenthesis to indicate that it is an object.
- The bottom area contains the values of the attributes for this particular instantiation of the class. In this case the name of the attribute is included for clarity, however, convention allows that the name need not be included if there is no chance for confusion.

The dotted line between the class box and the object "round-tangle" indicates that the object represented by the round-tangle is a member of the indicated class. In a practical sense, the class represents the abstraction of the operations (logic, algorithms, code, etc.), and of the attributes (variables definitions, constants definitions, etc.) that belong to that class of objects. The object represents the allocation of data values to the attributes. The unique set of values for a particular object is what makes it a unique and identifiable member of the class. For example, Instrument

Type will have a unique cal. factor which could alone distinguish it as a unique instrument type. The compiler or operating system normally assigns a unique value to a "object handle attribute" . This handle is invisible to the developer and is used to account for the case where two or more objects in the same class have the exact same values for all of the attributes. For example, a used car dealer may track each car only by make, model, year, two/four door and condition. There are thousands of Chevy, Malibus, 1988, two door, in good condition. The compiler will apply a handle to each of the objects of the class in order that each may remain unique.

N.3.6 Links and Associations

Objects will have relationships with other objects. We must provide a means for describing the relationship that a particular object has with any other object from which it needs service or to which it provides service. These relationships are called links when they refer to the relationship between two object (instantiations) and associations when they refer to the relationship between two classes.

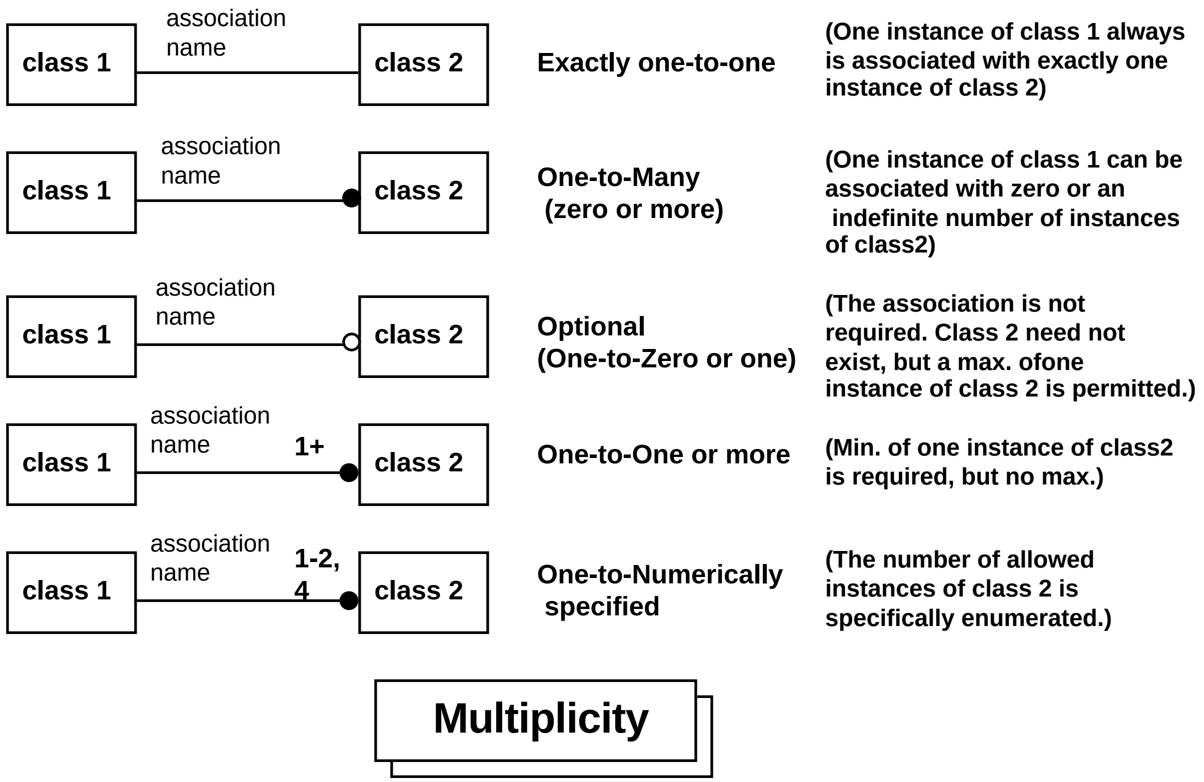
So, a link is a physical or conceptual connection between object instances (e.g. Joe works-for HAIS. Joe and HAIS are specific objects/instances, and "works-for is the link). **An association describes a group of links with common structure and common semantics** (e.g. Person Works-for Company). Person and Company are classes of which Joe and HAIS are object/instances, and Works-for is the association. Associations are often implemented as pointers or references from one class to another.

N.3.6.1 Standard, No-Frills Associations, and Multiplicity

The standard association is represented on an object diagram as a line between the two participating classes. The line may or may not be labeled. Labels are optional notation that are added for clarity of the model.

Not all associations between classes are one-to-one, therefore the model notation provides the facility to indicate "multiplicity" in the association. Multiplicity specifies how many instances of one class can relate to a single instance of an associated class. Multiplicity constrains the number of related objects.

The standard Object Modeling Technique (OMT) "Multiplicity Ball" notation is displayed in Figure N.3.6.1-1.



Exactly one-to-one (One instance of class 1 always is associated with exactly one instance of class 2)

One-to-Many (zero or more) (One instance of class 1 can be associated with zero or an indefinite number of instances of class2)

Optional (One-to-Zero or one) (The association is not required. Class 2 need not exist, but a max. ofone instance of class 2 is permitted.)

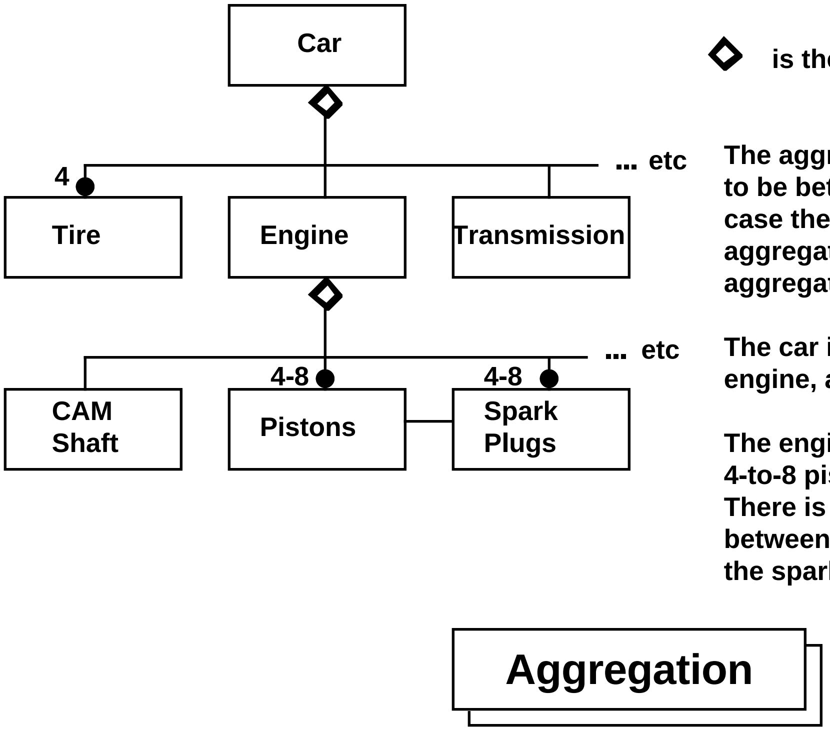
One-to-One or more (Min. of one instance of class2 is required, but no max.)

One-to-Numerically specified (The number of allowed instances of class 2 is specifically enumerated.)

Figure N.3.6.6-1. Standard OMT Multiplicity Bell

N.3.6.2 Aggregation

Aggregation is referred to as the "part-whole" or "a-part-of" relationship. For example, a car is the whole or the overall assembly, while the tires, engine, transmission, etc. are the parts. Therefore the tires, etc. are "a-part-of" the car assembly. In turn the parts can be aggregations of their constituent parts. For example, a piston, CAM shaft, spark plugs a "a-part-of" the engine. Multiplicity may be added to the aggregation, e.g. four tires per car. So, as shown in Figure N.3.6.2-1., **Aggregation is the association in which the relationship is between the whole and its parts**.



◇ is the aggregation symbol

The aggregation relationship is said to be between only two classes, in this case the car-tire accociation is an aggregation, the car engine is an aggregation, etc.

The car is made up of four tires, an engine, and a transmission.

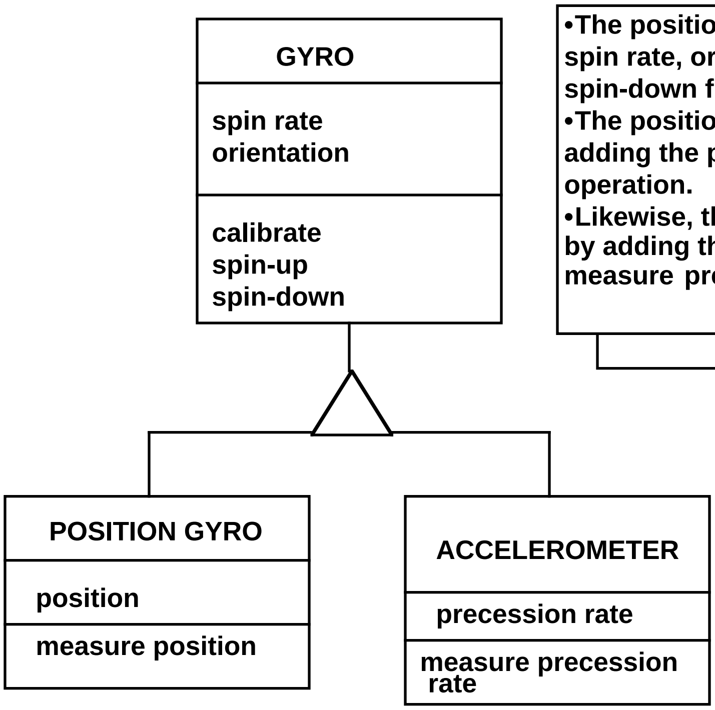
The engine is mae up of a CAM shaft, 4-to-8 pistion, and 4-to-8 spark plugs. There is a one-to-one association between instances of the pistons and the spark plugs.

Figure N.3.6.2-1. Aggregation

N.3.6.3 Generalization and Inheritance

As shown in Figure N.3.6.3-1., **Generalization and Inheritance are abstractions for sharing similarities among classes while allowing the classes to preserve their differences**. Generalization is the relationship between a class (**superclass**) and one or more refined versions (**subclass**) of it whereby the versions retain an identity as being one of the superclass yet distinguished from other subclasses. Attributes and operations common to the group of subclasses are attached to the superclass and shared by each of the subclasses.

For example, if motor vehicle is a superclass then truck and car are subclasses. Both types have many attributes and operations in common, but the fact that they differ in several attributes and operations distinguish them. The common attributes and operations would be attached to the vehicle superclass while the attributes and operations unique to truck and car would be attached to the truck and car subclasses. Each of the subclasses is said to **inherit** the features of the superclass. The superclass is said to be the **ancestor** of all of its subclasses, and the subclasses are said to be the **descendants** of the superclass.



- The position gyro and accelerometer both inherit spin rate, orientation, calibration, spin-up, spin-down from the gyro superclass.
- The position gyro subclass is specialized by adding the position attribute and the measure position operation.
- Likewise, the accelerometer subclass is specialized by adding the precession rate attribute and the measure precession rate operation.

Generalization and Inheritance

Figure N.3.6.3-1. Generalization and Inheritance

This document has not yet been approved by the Government for general use or distribution.